Simulation and Modeling of a New Medium Access Control Scheme for Multi-Beam Directional Networking

Brian Proulx, Greg Kuperman, Nathaniel M. Jones, Thomas Goff
MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02420
{brian.proulx, gkuperman, njones, thomas.goff}@ll.mit.edu

***Abstract*—In this paper, we analyze a new medium access control (MAC) protocol for multi-beam directional network via high-fidelity simulation using a real-time emulator. Multi-beam directional systems are a novel approach to networking which leverage recent advances in physical layer technology, allowing formation of multiple simultaneous beams in both transmit and receive. These multiple beams significantly reduce the burdensome coordination between the transmitter and receiver and enable an uncoordinated, distributed random access scheme that offers high throughput. This paper is the first to characterize the performance of such systems using real-time simulation tools. In addition to implementing the random access scheme, several novel MAC features are developed that allow for robust communication, such as location tracking and extrapolating neighbor transmit or receive state.**

For this paper, we implement our protocol in both simulation and a new Extendable Mobile Ad-hoc Network Emulator (EMANE) model that allows for real-time, high fidelity performance evaluation. Using EMANE allows us to better understand the performance of the newly developed protocols by running real-time applications through the network. We show that our results from the EMANE model and simulator coincide with the theoretical network throughput, which allows for empirical characterization of scenarios in which theoretical results have not been derived. Furthermore, through our work, we stress test the EMANE model and quantify the maximum number of nodes that we can operate in real-time. Through this testing, two bottlenecks are identified: 1) infrastructure issues, where the amount of data passed between the servers is too high, and 2) computation issues, where calculating the interference on the packets becomes too time consuming.

TABLE OF CONTENTS

1. INTRODUCTION 1
2. MAC PROTOCOL 2
3. DISCRETE EVENT SIMULATOR 4
4. EMANE MODEL 4
5. RESULTS 7
6. CONCLUSION 7
REFERENCES 8

1. INTRODUCTION

Moving from omnidirectional to directional systems offers a host of advantages, such as increased communication range and higher data rates [1]. Traditionally, directional signals are generated by mechanically steering an antenna. However, a new technology, the digital phased array (also called a fully digital antenna array) allows for significant flexibility in directional communications [2]. A phased array is an array of antenna elements, each with an analog to digital converter behind it. When receiving, the received signal is fully digitized, which allows receive beams to be generated *a posteriori* by processing the received energy. Essentially, the ability to post-process allows all directions to be searched simultaneously and the direction with the best signal to noise ratio (SNR) to be selected. Another key advancement from this underlying technology is the ability to form multiple simultaneous receive or transmit beams. The capability to adaptively form transmit and receive beams can greatly enhance communication systems and has already been applied to radar systems [3]. In particular, the ability to post-process to receive many signals at the same time greatly reduces the complexity at the medium access control (MAC) layer, and the network throughput can be substantially higher by transmitting concurrent, independent streams.

This work centers on exploiting this new capability to create a new distributed, low complexity, random access MAC protocol for multi-beam directional networks. Many previous works have addressed MAC layer design for directional networks, but did not consider phased arrays to *a posteriori* form receive beams [4]. This leads to complex MAC schemes designed to schedule transmissions between nodes, so that the receive beam would be pointed at the transmitter in order to receive the packet [4]. These schemes result in reduced network throughput and a loss of spatial reuse. Also, many protocols required request to send (RTS) and clear to send (CTS) handshaking to create a directional network allocation vector (DNAV) for virtual channel sensing [5] [6]. Clearly, random access schemes are not practical if tight transmitter-receiver coordination is necessary for packet reception.

The airborne domain is the focus of this paper, and note that both military and civilian networks benefit from the decreased probability of detection and robust anti-jamming capabilities of directional networks. In this scenario, there can be very long distance links between nodes. Thus, propagation delay is usually far longer than the time needed to transmit a packet. Therefore, the previous schemes requiring either virtual or actual channel sensing are impractical, as most of the time spent would be used to reserve a channel [7]. However, this same issue can arise in traditional high rate communication systems as well. For example, at 10 Gbps, a 1 Kb packet is transmitted in 100 ns, which is equivalent to a propagation delay of 30 meters. Additionally, this domain also serves to differentiate the underlying physical layer from massive MIMO (multiple input, multiple output), because in this case there is very little scattering, but on the ground in massive MIMO, there is a rich scattering environment. Massive MIMO centers around using a highly capable base station to transmit multiple streams to a set of receivers. Channel state information is fed to the trans-

This work is sponsored by the Assistant Secretary of Defense Research and Engineering via Air Force contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

mitter in order to form beams to each receiver [8]. In contrast, the airborne domain does not require accurate channel state information, though it would only improve performance, as the angle of transmission is the line of sight path, which can be calculated from location information. Additionally, in our MAC protocol, it is assumed that all nodes will have antenna arrays and communicate amongst themselves, rather than single antenna systems communicating only with a single base station equipped with a digital phased array.

The new MAC protocol presented in this paper introduces many features in order to leverage the advancements of the underlying physical layer technology. Instead of relying upon a synchronized time slotted system, an unsynchronized time slotted, random access scheme is implemented. At a glance, each node probabilistically transitions between periods of transmission and reception, independent of all other nodes; further details of this scheme can be found in [9]. With the random access scheme, each node is able to track the transmit or receive state of its neighbors to ensure that the transmission arrives while neighbor is in a receive state, greatly reducing the number of failed transmissions (and interference) in the network. Another MAC feature is a simple neighbor discovery protocol to create new links in the network. Node location tracking, as precise node positions are necessary for pointing a transmit beam in the correct direction as well as selecting the correct transmit power, is a key MAC capability. Power control is also needed to transmit multiple beams simultaneously, as the antenna array is constrained in its total power. In order to reduce interference, the entire bandwidth is divided into disjoint channels, and a method for choosing a channel for communication is developed. Combined with channel selection, checking to ensure that a node does not interfere with itself is another feature: when transmitting multiple beams, if two receivers are not separated enough in angle, then the two transmissions would collide with each other.

In order to properly test this new protocol, we developed a custom discrete event simulator in C++, and a new Extendable Mobile Ad-hoc Network Emulator (EMANE) [10] model. These tools are used to both quantify the network throughput as well as implement new approaches to improve the system. The discrete event simulator is a custom C++ program that uses the MAC features and physical layer signal-in-space model from the EMANE model, but does not run at real time, and cannot carry real traffic. The simulator is useful for further understanding the underlying behavior of the system, as large amounts of varying data can be extracted. Also, developing new features is significantly easier than for EMANE, and this allows for fast prototyping. The simulator is completely centralized, and for small number of nodes runs significantly faster than real time. As the number of nodes increases, the run time of the simulator transitions to slower than real time.

System emulation was performed by developing a new EMANE model. The EMANE model is a close representation of the actual system as real traffic is sent and received, and the MAC features run at real time. At the physical (PHY) layer, interference is calculated on each packet at a high fidelity, which is necessary for emulating directional networks. However, this high fidelity interference calculation is computationally expensive and limits the total number of nodes that can be emulated at once. In addition to the EMANE model, we further extended the capabilities of the Common Open Research Emulator (CORE) [11], allowing for robust distributed emulation in a portable fashion. Python scripts that interface with CORE allow for quick set up of repeated tests and easy data collection. With the EMANE model and distributed emulation, the performance of the model could be analyzed. Specifically the most challenging case, backlogged all-to-all traffic, was tested. In this case, the maximum number of nodes possible was determined to be 11, due to both infrastructure issues, the amount of data sent between the servers, and computational issues, the time needed to compute the interference on each packet.

In summary, we designed a new MAC protocol to exploit the new underlying physical layer capabilities. This protocol utilizes random access and is designed with robust location tracking and neighbor time slot tracking in order to dramatically reduce dropped packets and interference. We implemented this protocol in our simulator as well as our EMANE model, and determined the major fundamental bottlenecks that restrict the EMANE model to a maximum of 11 nodes. We used these tools to show that the theoretical, simulation, and emulation results coincide, which allows our new tools to numerically characterize scenarios for which there are no theoretical results, namely all to all traffic.

The paper is organized as follows: first the MAC layer is developed. Next, the simulator is covered, and then the EMANE model and distributed emulation capabilities are presented. Next, the limitations of the EMANE model are discussed and quantified. Following that, sum network throughput results are shown, and finally, we summarize the contributions of this paper.

2. MAC PROTOCOL

In order to fully utilize the new capabilities in a multi-beam directional system, we developed a new MAC protocol. We examine this in detail by outlining its features, beginning with the basic random access scheme we previously developed [9]. Next, new features that were designed and implemented are presented, namely time slot tracking of neighbors, neighbor discovery, location tracking, power control, channel selection, and self-interference checking. These features highlight the new challenges of a multi-beam directional system, namely a carefully constructed access scheme to allow for high spatial reuse when transmitting to multiple neighbors simultaneously and the importance of location information (for pointing beams in the correct direction).

Access Scheme

We begin with a brief description of our access scheme from [9]. In this protocol, time is slotted, but the time slots across nodes are not synchronized. Thus all nodes use a fixed time slot for transmission or reception, but the starts of these slots are not aligned in time. Each time slot is designated either a transmit or receive time slot, as transmit-while-receive is not possible in this system. There is a single integer system parameter, denoted x, which is used to transition between transmit and receive time slots. The time slots are assigned to either transmit or receive as follows: a node will have x receive time slots consecutively. After those slots, there is a single transmit slot, and the slot after that is chosen probabilistically. With probability $\frac{1}{x}$, the process begins again (starting with x receive time slots), but with probability $\frac{x-1}{x}$, this next slot is a transmit slot; after this next transmit slot, the same random process is performed, continuing until the realization is x receive slots. In short, the time slot realizations are x receive slots followed by one

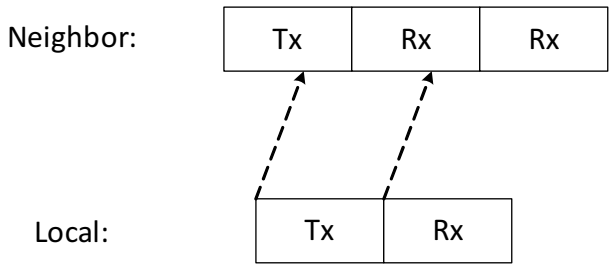


Figure 1. Time slot tracking example

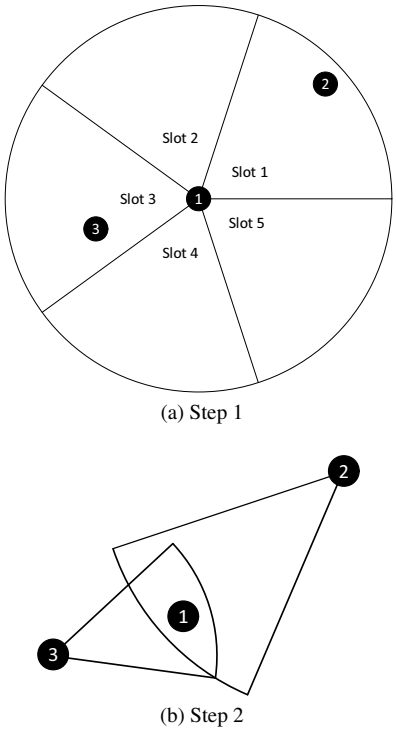
transmit slot, followed by between zero and ∞ additional transmit slots, which then transition back to the beginning of the process with x receive slots. For all results in this paper, $x = 5$, which results in five receive slots followed by a few transmit slots, as a typical realization.

Time Slot Tracking

Note that under this access scheme, learning the realizations of neighbor's time slots (either transmit or receive) will allow a node to refrain from transmitting a packet to a neighbor if that neighbor is currently transmitting. This drastically reduces the number of dropped packets in the system, because if a node transmits to a neighbor that is in the transmit state, then the packet will not be received. Additionally, refraining from transmitting to nodes that will not receive the packet results in less interference for other concurrent packets. Thus, each node tracks the state of its neighbors' time slots to ensure that the neighbor will be in a receive state for the duration of the packet. To allow this ability, each node uses a pseudo-random number generator to determine its time slot state. The seed of this generator determines the resulting state. In our system, a node attaches its random seed and the number of times this generator has been used to the header of the packet for each transmission. A receiver can then determine the start of this time slot (by subtracting out the propagation time), and from this information, calculate the entire future of the node's time slot states. Thus, a node will only transmit to a neighbor if that neighbor is in a receive state, otherwise a packet is not sent. This reduces the amount of dropped packets, the amount of interference, and saves aperture power for transmissions that will be successful to other neighbors. See Figure 1 for an example. In this example, the local node will not transmit a packet to the neighbor because the neighbor is in a transmit slot for the first part of the packet. The angle of the dashed lines indicates the propagation delay of the packet from the local node to the neighbor node.

Neighbor Discovery

An important feature of the MAC protocol is neighbor discovery. As a directional system cannot broadcast in all directions simultaneously, the only alternative is to sweep over the full 4π steradian space to discover other nodes. The fundamental challenges of this problem are outlined in [12]. In the EMANE model, this search is currently limited to a 2π search in the azimuth direction only. Each node subdivides the space by the beam width, and transmits a packet in each direction, until the full 2π space is covered. This packet contains the node's location as well as the period in time when it will be finished sending neighbor discovery packets. When a neighbor receives one of these packets, it waits until the end of the transmit time and then responds with its own hello packet, containing its own location, as well as that of all of this neighbor's neighbors. For an example, see Figure 2. In this figure, node 1 sends out five packets sequentially, and



(a) Step 1

(b) Step 2

Figure 2. Neighbor discovery process

after these transmissions, nodes 2 and 3 respond with their own hello packet.

Location Tracking

Another important feature is location tracking. Due to node mobility, it is vital that each node tracks the locations of its neighbors. Otherwise the transmit beam may not be pointed correctly, and the neighbors will no longer be able to communicate. Nodes exchange location information periodically, including their velocity, to ensure that beams are pointed in the correct direction. Additionally, each node stores all locations of its neighbors, and before transmitting a packet, the node will use dead reckoning to extrapolate the neighbor's position from the last received location and velocity information in order to determine the direction to point the transmit beam. Also, as a node knows the last position and velocity it sent to its neighbor, it can create that neighbor's dead reckoning estimate of its own position. Based on its current position and velocity, the node can use this to check if it will still be within the neighbor's transmit beam. If the node has deviated too much from its previous location and velocity, it will update its neighbor. Specifically, if the node is beyond $\frac{1}{4}$ beam width away from its dead-reckoned position, it will update the neighbor with its new position and velocity. For example, in Figure 3, the dead reckoning position of node 1 is shown as node 1a, and its actual position is node 1b. As its actual position is more than $\frac{1}{4}$ beam width outside of node 2's dead reckoning of node 1, represented by the shaded area, node 1 will send a position update to node 2. Though this figure is two-dimensional, this process occurs with respect to both the azimuth and elevation angles of the transmit beam.

For three dimensional location tracking, the dot product of the two vectors (one vector being from the neighbor to the extrapolated position, the other being from the neighbor to the

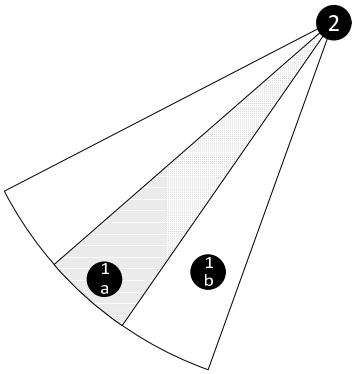


Figure 3. Location tracking example

actual position) is used to calculate the difference in the two angles. If this dot product is larger than a quarter beam width minus the guard band, then an update is required. Note that this requires the beam width to be equal to the beam height, in order for the beam to be a circular cone in three dimensions. This guard band, measured in degrees, ensures that this check is performed with enough time to transmit a packet to the neighbor. If the check was precisely one quarter beam width, then the resulting transmission may not reach the neighbor in time. Additionally, using one quarter beam width ensures that the deviations of both neighbors' positions are considered.

Power Control

Power control is a key feature in the MAC layer. Each node knows its neighbor's location and the minimum signal to interference plus noise ratio (SINR) needed for the packet to be correctly decoded at the receiver. Thus, calculating the pathloss of a packet allows for the transmit power to be scaled appropriately. As each aperture has a maximum power, correctly scaling the transmit power allows for more simultaneous beams, as well as decreases the amount of interference on surrounding transmissions. Additionally, a multiplicative guard factor (greater than one) is used to scale the transmit power to make each transmission more robust to interference as well as slight errors in beam pointing (which result in decreased gain).

As nodes are mobile, there is a multiplicative factor when selecting transmit power, which accounts for the fact that while the node will still be in the transmit beam in terms of angles, it may be farther away than the dead reckoning estimate. If the power is selected as the minimum necessary to close the link, any deviation in the distance will result in a dropped packet. In order to select this multiplicative factor, consider an example from our EMANE model. In EMANE, the pathloss in dB (using the freespace propagation model) is

$$20 \log_{10} \left(F \frac{\text{frequency (Hz)}}{10^6} \frac{\text{distance (meters)}}{1000} \right), \quad (1)$$

where F = 41.916. We set the maximum allowed error in distance to 300 meters, which results in a maximum propagation delay error of approximately one microsecond. Using a center frequency of 22 GHz and considering a 300 meter location error, if the nodes are 5 km apart, the difference in received power is about 0.5 dB, and at 50 km of separation, the difference is 0.1 dB. In order to counteract this loss, the multiplicative power guard factor is set to 1.34, meaning the transmit power (in mW) is scaled up by this amount. This scaling results in a 6.5 dB received power, if the location information is accurate and there is no interference. As the EMANE model assumes that the minimum power to correctly decode a packet is 5.24 dB, this results in 1.26 dB of clearance for location errors.

Channel Selection and Self-Interference Check

Channel selection and checking for self-interference are both used to ensure that two simultaneous transmissions from a node will not cause enough interference that the other transmission would not be received. In each time slot, the list of a node's neighbors is randomly ordered, and this list is used to schedule transmissions. For each neighbor, a random channel is selected, and then all of the other previously scheduled transmissions on this channel are examined to see if this potential transmission would interfere in the main beam. If the potential transmission does interfere, a new channel is selected and checked. If this potential transmission is unable to be sent on any channel, it is dropped.

3. DISCRETE EVENT SIMULATOR

An important tool in developing and analyzing this system is the discrete event simulator. Rather than real-time emulation like EMANE, the discrete event simulator is a completely centralized piece of software that computes the link utilization for all nodes over a time period. When the number of nodes is low, the simulator runs much faster than real time, but when the number of nodes is high, the simulator can take significantly longer than real time. The primary purpose of this tool is to allow for a simple, flexible simulator that allows for easy implementations of new features as well as the ability to extract more data than is possible with the EMANE model. Rather than actually transmit packets, the simulator calculates the final SINR of transmissions, and reports the overall link utilization.

The simulator is a custom C++ program, written from scratch. It uses the same interference and power calculations as EMANE to maintain compatible results. Additionally, the key MAC features, such as power control, channel selection, and the self-interference check are implemented to model the real system. The flexibility of this tool allows for interesting results, such as the effect of the minimum SINR to correctly receive a packet on the overall network throughput. This ability to gain rapid insight into MAC issues without the heavy cost of developing and testing new EMANE features fills a capability gap in the development of the overall system.

4. EMANE MODEL

In this work, a new EMANE model was developed that includes both the MAC and PHY layers of the design, using EMANE version 0.9.3. This MAC layer includes implementations of the features described in Section 2. Along with the MAC layer, a PHY layer was implemented in EMANE, which performs the receive beam pointing, determines the transmit and receive gains, and calculates the interference.

PHY Model

In order to fully implement this system, a model of the PHY layer was written in EMANE. In a real system, the PHY layer would be responsible for many things, such as modulation, coding, and interleaving, but as EMANE is a packet-level simulator, the EMANE PHY layer is responsible for modeling the signal in space. That is, it calculates the

transmit and receive beam gains, the propagation delay, the path loss, and importantly: the receive beam pointing and interference modeling, i.e., determining the SINR of each packet.

As previously mentioned, the real system will be able to process all receive directions simultaneously by utilizing large-scale parallel processing. This is implemented in the model by having the receiver point a receive beam precisely in the direction of the transmitter. This represents the ability to choose the direction that results in the maximum SNR from all possibilities. However, a real system may have small errors in pointing a receive beam, but this is not modeled here.

The interference model for the EMANE PHY layer, also known as the signal in space model, is very detailed compared to traditional EMANE models. As there are multiple simultaneous transmissions occurring on different channels in different directions, a thorough accounting for interference is needed.

An an example, focus on a single packet being received at a node. This packet has a receive beam associated with it, which is pointed directly at the transmitter. All other packets on the same channel are considered interference, and all packets on other channels do not interfere with this packet. For each packet on the same channel, the transmit gain is calculated, based on the direction of transmission relative to this node. Next, the receive gain is calculated using the receive beam pattern for this packet. Then the interference from this other packet is calculated on the packet of interest, and scaled linearly by the amount of time the two packets overlap at the receiver. Thus, each receiver calculates the interference for each packet by considering every other packet in the network. After calculating the final SINR of a packet, if the SINR is above a threshold (5.24 dB in this model), it is passed up to the MAC layer as a correctly decoded packet. This final step could be easily extended to a packet-correct curve (PCR), which translates the SINR to a probability of reception, in future work.

Treating all other packets as interference and calculating these values for every packet is very computationally demanding. There are a few modifications that we developed to reduce the computational load. First, if the packet is not destined for the node, then the node does not calculate its SINR; it only uses that packet as interference for packets destined for this node. Also, the SNR (with no interference) is calculated for each packet destined for the node, and if it is below the threshold (5.24 dB) then its interference is not calculated, as it will be dropped regardless. A thorough discussion of the characterization of this computation challenge is presented later in this section.

Distributed Emulation

In order to emulate a larger number of nodes, the emulation was set up to run in a distributed fashion across multiple servers. As interference modeling is computationally expensive, running only two nodes on each server greatly increases the number of nodes possible in a test. We developed new CORE python scripts to automate testing and data collection.

The CORE python framework allows for a programmatic method for setting up emulation experiments. Rather than hard code all of the underlying pieces for EMANE (such as the over-the-air (OTA) channel, generic routing encapsulation (GRE) tunnels between the servers, writing every XML node file, among others), the CORE python framework constructs these in a python script that is portable between servers and underlying network setups. While this framework is robust for running EMANE on a single server, significant work was necessary for running EMANE in a distributed manner.

Performance Characterization

In order to understand the limitations of our EMANE model and distributed environment, we performed a series of tests to show its performance as a function of the number of nodes. The key challenge underlying emulating large numbers of nodes is that the spatial reuse of a directional system, combined with multi-beam transmission, allows for a high theoretical network throughput, which is difficult to emulate in real time. After some initial study, we found that two key factors limit the number of nodes: infrastructure and computation. For the infrastructure piece, each packet transmitted by a node must be transmitted to all other servers, which in turn pass this packet to their nodes. However, the large amount of data to be sent causes the network connecting the servers to become overloaded. The computation challenge is that calculating the interference for each packet is more difficult with more packets being transmitted, and the number of packets scales with the number of nodes.

It is important to note that this performance characterization is for a very demanding scenario. We consider the all to all traffic pattern, in which each node has a backlogged flow to each other node in the network. The total number of flows in a network with n nodes is $n(n-1)$, which scales quadratically with n. While this traffic pattern may not model realistic scenarios, it does represent an upper bound to the emulation ability. This testing was performed with a minimum of two nodes per server, and therefore if n is odd, one server hosts three nodes.

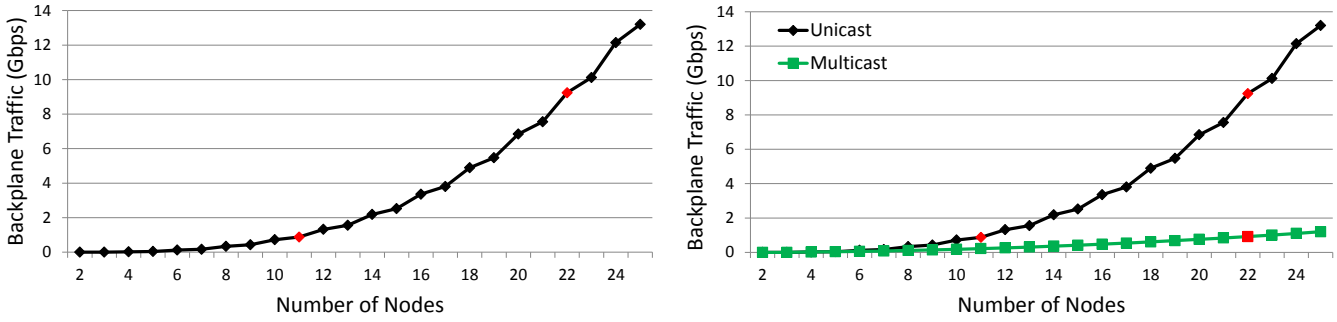
For these tests, there are two testbeds. One testbed is comprised of 19 identical servers, each with a quad core processor, connected together on a switch with 1 Gbps ethernet. The other testbed is 8 machines, each with dual 14-core processors, connected via a switch with 10 Gbps ethernet.

Infrastructure— To start, we focused on the infrastructure challenge. The physical network connecting the servers is called the backplane, and in EMANE, each packet transmitted by a node must be distributed to all of the other servers over the backplane. In our configuration, the CORE python framework unicasts each over-the-air packet to each server, so a single EMANE transmission becomes $(m-1)$ packets over the backplane, where m is the number of servers. Thus, the backplane traffic can be calculated as the flow rate times the number of flows times the number of servers minus one:

$$rn(n-1)(m-1), \tag{2}$$

where r is the effective link rate. As an example, consider a 2 Mbps effective link rate, with 11 nodes distributed over 5 servers. This results in 0.88 Gbps of backplane traffic. A graph of the backplane traffic assuming 2 Mbps effective link rate can be seen in Figure 4(a). Note that this figure shows that only 11 nodes are possible with a 1 Gbps ethernet backplane, and 22 nodes are possible with a 10 Gbps backplane. If the CORE python framework is changed to use multicast between the servers, instead of unicast, the $(m-1)$ term is dropped from Equation 2, and the resulting performance is plotted against the unicast performance in Figure 4(b). Notice that now 22 nodes can be emulated using a 1 Gbps ethernet backplane.

In order to test the infrastructure performance, two metrics



(a) Backplane traffic for unicast

(b) Backplane traffic for multicast compared to unicast

Figure 4. Backplane traffic

were considered. The first is the number of missing packets. For each test, every transmitted packet is tracked to ensure that it is received at each EMANE node. Note that data is collected before entering EMANE and deciding that the packet is received or dropped; that is, every packet should be received at this point. The results can be seen in Figure 5(a). Note that with a backplane of 1 Gbps, beyond 11 nodes, the theoretical maximum, the number of missing packets increases dramatically. The 10 Gbps backplane is able to serve all of the packets without issue.

The next metric is the delay for each packet. Again, this is measured for every transmitted packet, and represents the time between the packet leaving the transmitter EMANE, and when it first arrives at all receiver EMANE models. The results in Figure 5(b) show similar results to the missing packet metric. Below the theoretical maximum for 1 Gbps ethernet, the delay is very small, but above that number, the backplane is overloaded and the delay skyrockets. Again, the 10 Gbps testbed is well below the theoretical maximum and performs well.

Computation—Along with the infrastructure challenge, there is a significant computational challenge. That is, each node must calculate the interference for every packet. With more nodes in this all to all traffic scenario, there are more packets, and thus more interference calculations to perform on each packet. To gain some insight, consider a system with n nodes, where n is an even number for simplicity, and note that the maximum number of packets transmitted at a given time occurs when half of those nodes are transmitting, and half are receiving. This results in $\frac{n}{2} \cdot \frac{n}{2}$ total transmissions. Thus a receiver calculating the interference of a single packet would see $\frac{n}{2} \cdot \frac{n}{2} - 1$ interfering packets. So in total, there are

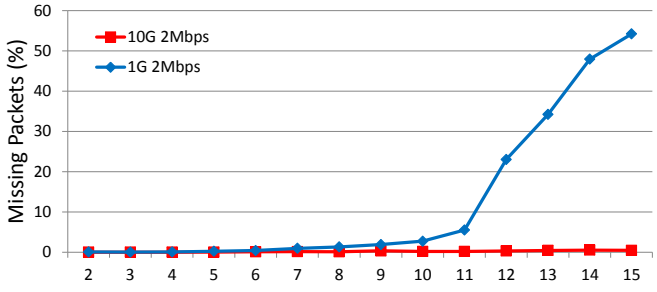
$$\frac{n}{2} \cdot \frac{n}{2} \left(\frac{n}{2} \cdot \frac{n}{2} - 1 \right) \tag{3}$$

packets to compute interference on at each node. This quantity scales as n^4. One of our improvements to the interference calculation is to compute interference only on packets destined for the node, which decreases the number of packets to compute interference for, from $\frac{n}{2} \cdot \frac{n}{2}$ to only $\frac{n}{2}$ packets, resulting in a scaling behavior of n^3. While better, this is still a significant amount of computation.

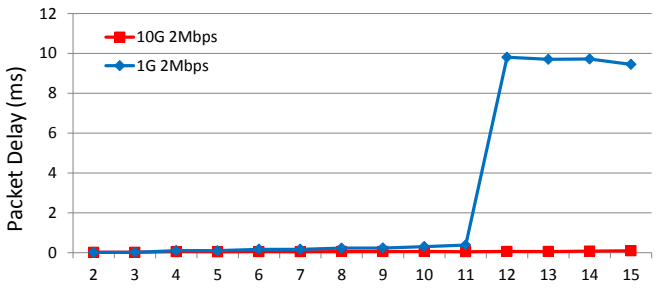
In order to understand the computation challenge, it is important to understand the EMANE process execution model. At start up, EMANE creates several threads. A few of these are for relatively lightweight tasks, such as logging or processing location events. But most importantly, the MAC layer and the PHY layer are launched as separate threads. So both of these layers can be run simultaneously on different cores, but within a layer, there is only a single thread. Consequently, although there may be multiple simultaneous packets being received, they are processed serially. We will see that this drastically limits the number of nodes that can be run, even assuming that a single thread could run on a core without any context switching. This assumption does not hold in the quad core case, as the MAC and PHY layers for each node will be switched periodically with the other EMANE threads.

In order to gain insight into the computational challenges, we used the Systemtap software. This software compiles a kernel module that profiles applications as they are run. This has the advantage of not requiring any changes to the EMANE code and allows it to run in real-time, which is vital for accurate results. After running, the duration of the functions called can be computed and the overall computational load can be analyzed. The following plots demonstrate the time taken by each thread as a fraction of the total time. That is, during the interval of time monitored, this thread used a certain percentage of the time. This is the "Actual" labels for the graphs. Also plotted is the extrapolated percentage of time for this thread. This is calculated by multiplying the duration of each function called in the packet chain by the expected number of packets received in an interval. A value greater than 100% represents that the thread does not have enough compute ability to process all of the packets it is receiving. There are two test beds for these tests: servers with quad core processors, and servers with 28-core processors. The results for the MAC layer thread are seen in Figure 6(a), and the PHY layer thread is shown in Figure 6(b). These highlight that even the theoretical maximum number of nodes (achieved if each thread is run uninterrupted on a core) is limited by the PHY layer thread. Examining Figure 6(b) shows that this limit is eight for the quad core servers, and ten for the 28-core servers. Another interesting note is the relative similarity between the two testbeds. Though the 28-core servers perform better, the performance is limited by the fact that each layer runs in a single thread. Despite being able to run 28 threads simultaneously, the performance is not markedly better than that of the quad core machines.

Characterization Results—The overall performance of the system using all to all traffic with backlogged queues can be visualized by looking at the sum network throughput. By summing the throughput of each link, the performance of EMANE can be compared against the relative ground truth of the simulator results. Results with a 2 Mbps effective link rate can be seen in Figure 7(a). For both testbeds, the EMANE results overlay the simulation results closely, but begin to diverge as the number of nodes reaches the maximum possible. For the quad core servers, the theoretical

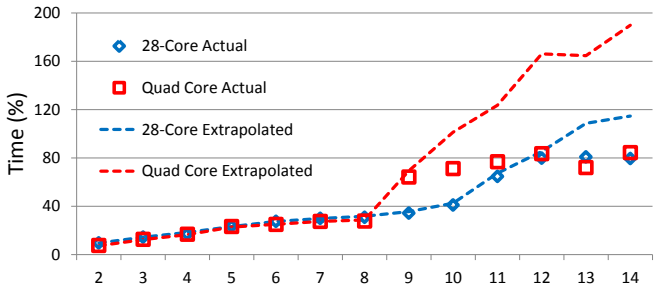


(a) Percentage of missing packets

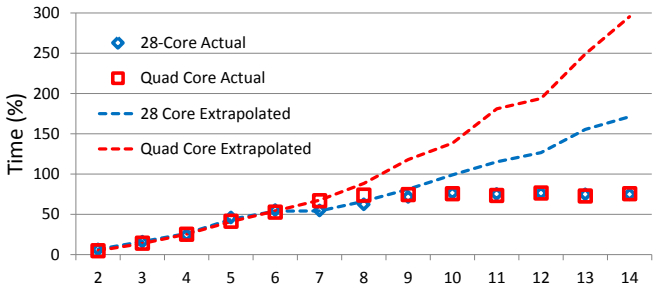


(b) Packet delay

Figure 5. Infrastructure performance metrics



(a) MAC layer computation



(b) PHY layer computation

Figure 6. Computation performance metrics

maximum of the infrastructure is reached at 11 nodes, and the overall network throughput drops because most of the packets are dropped between servers. Similar behavior is seen in for the 14-core testbed, but rather than the infrastructure issues limiting the network throughput at 11 nodes, it is the computation issues in the PHY and MAC threads.

It is important to point out that these limits are highly dependent on the length of the time slot and the size of packets. If the overall number of packets transmitted is decreased by a factor of 10, then the EMANE model is still a good representation of the system. See Figure 7(b) for an illustration of system performance using an effecitve link rate of 200 kbps. Notice that the limit for the infrastructure and computation challenges is well beyond the number of nodes plotted.

5. RESULTS

To demonstrate the overall system, sum network throughput results were generated. These give insights into the overall system performance as a function of the number of neighbors. The node laydowns vary slightly between the all to one case and the all to all case. In the all to one case, a single node is placed in the center of a circle, and n nodes are distributed uniformly at random around this central node. Each of these nodes then transmits a backlogged flow to the center node. In the all to all case, nodes are distributed uniformly at random in a square area, and every node transmits a backlogged flow to every other node.

All to One

For the all to one case, there are theoretical results, EMANE results, and discrete event simulator results. The theoretical results are generated from Equation 23 (utilizing the forms from Equations 24 and 25) in [9]. In Figure 8, these results are shown for the number of neighbors spanning 1 to 30. All three models of the system coincide, showing that both the discrete event simulator and EMANE model are valid.

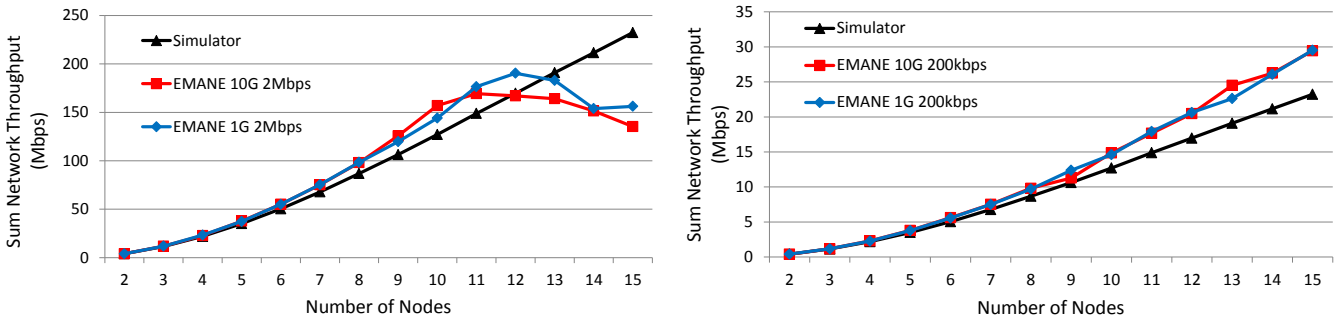
All to All

For all to all traffic, there are no theoretical results to compare against, but the simulation results represent the ground truth, as they aligned with the theoretical results for the all to one case, and the simulator does not have the infrastructure or computation issues that EMANE does. Figure 7(a) and Figure 7(b) show the network sum throughput.

6. CONCLUSION

In this paper, we describe our new MAC layer protocol that exploits the novel underlying physical layer abilities of a digital phased array, specifically the ability to form receive beams *a posteriori* and to form multiple transmit or receive beams. We designed new location tracking and power control methods to ensure that the transmit beam is correctly pointed with the correct power. Also, we proposed a scheme to track the state of a neighbor's random access protocol in order to drastically reduce the number of dropped packets and interference in the system. These ideas were tested in a C++ simulator as well as an EMANE model, and the scaling limitations of the EMANE model were discussed and quantified. Additionally, it was shown that the theoretical results coincided with both the EMANE model and simulator results, which confirms the validity of the results. Knowning this, numerical characterizations of scenarios without theoretical results were presented to further characterize the system capability.

There are plenty of avenues for future research as well.



(a) Network throughput at 2 Mbps

(b) Network throughput at 200 kbps

Figure 7. All to all results

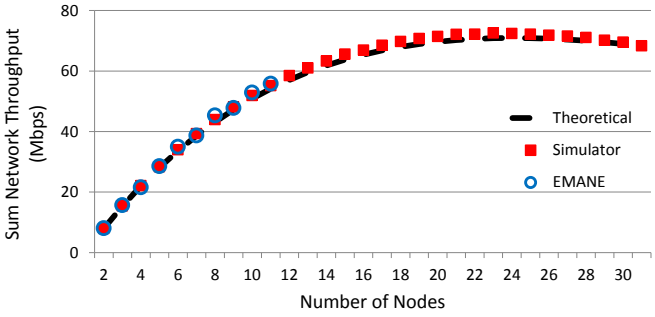


Figure 8. All to one results

Neighbor discovery in a full 4π steradian search space remains an open problem, and a more sophisticated location tracking algorithm could be developed that accounts for general platform movement abilities as well as predicted flight plans. Another path for research is contention resolution in dense networks, where neighbors are less than a beam width apart, as selecting the best set of links in a distributed manner is a difficult task.

REFERENCES

[1] S. Yi, Y. Pei, and S. Kalyanaraman, "On the capacity improvement of ad hoc wireless networks using directional antennas," in *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, ser. MobiHoc '03. New York, NY, USA: ACM, 2003, pp. 108–116.

[2] J. Litva and T. K. Lo, *Digital Beamforming in Wireless Communications*, 1st ed. Norwood, MA, USA: Artech House, Inc., 1996.

[3] M. Ascione, G. Bernardi, A. Buonanno, M. D'Urso, M. Felaco, M. G. Labate, G. Prisco, and P. Vinetti, "Simultaneous beams in large phased radar arrays," in *Phased Array Systems Technology, 2013 IEEE International Symposium on*, Oct 2013, pp. 616–616.

[4] H.-N. Dai, K.-W. Ng, M. Li, and M.-Y. Wu, "An overview of using directional antennas in wireless networks," *International Journal of Communication Systems*, vol. 26, no. 4, pp. 413–448, 2013. [Online]. Available: http://dx.doi.org/10.1002/dac.1348

[5] R. R. Choudhury, X. Yang, R. Ramanathan, and N. H. Vaidya, "Using directional antennas for medium access control in ad hoc networks," in *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '02. New York, NY, USA: ACM, 2002, pp. 59–70. [Online]. Available: http://doi.acm.org/10.1145/570645.570653

[6] T. Korakis, G. Jakllari, and L. Tassiulas, "A mac protocol for full exploitation of directional antennas in ad-hoc wireless networks," in *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, ser. MobiHoc '03. New York, NY, USA: ACM, 2003, pp. 98–107.

[7] D. Bertsekas and R. Gallager, *Data Networks (2Nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[8] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive mimo for next generation wireless systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 186–195, February 2014.

[9] G. Kuperman, R. Margolies, N. M. Jones, B. Proulx, and A. Narula-Tam, "Uncoordinated mac for adaptive multi-beam directional networks: Analysis and evaluation," in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, Aug 2016, pp. 1–10.

[10] (2016) Extendable mobile ad-hoc network emulator (emane). [Online]. Available: http://www.nrl.navy.mil/itd/ncs/products/emane

[11] J. Ahrenholz, "Comparison of core network emulation platforms," in *MILITARY COMMUNICATIONS CONFERENCE, 2010 - MILCOM 2010*, Oct 2010, pp. 166–171.

[12] M. E. Steenstrup, "Neighbor Discovery among Mobile Nodes Equipped with Smart Antennas," in *Scandinavian Workshop on Wireless Ad-hoc Networks*, May 2003.